Fig. 1
(Prior Art)

Fig. 2

Fig. 3

Fig. 4

5 / 34

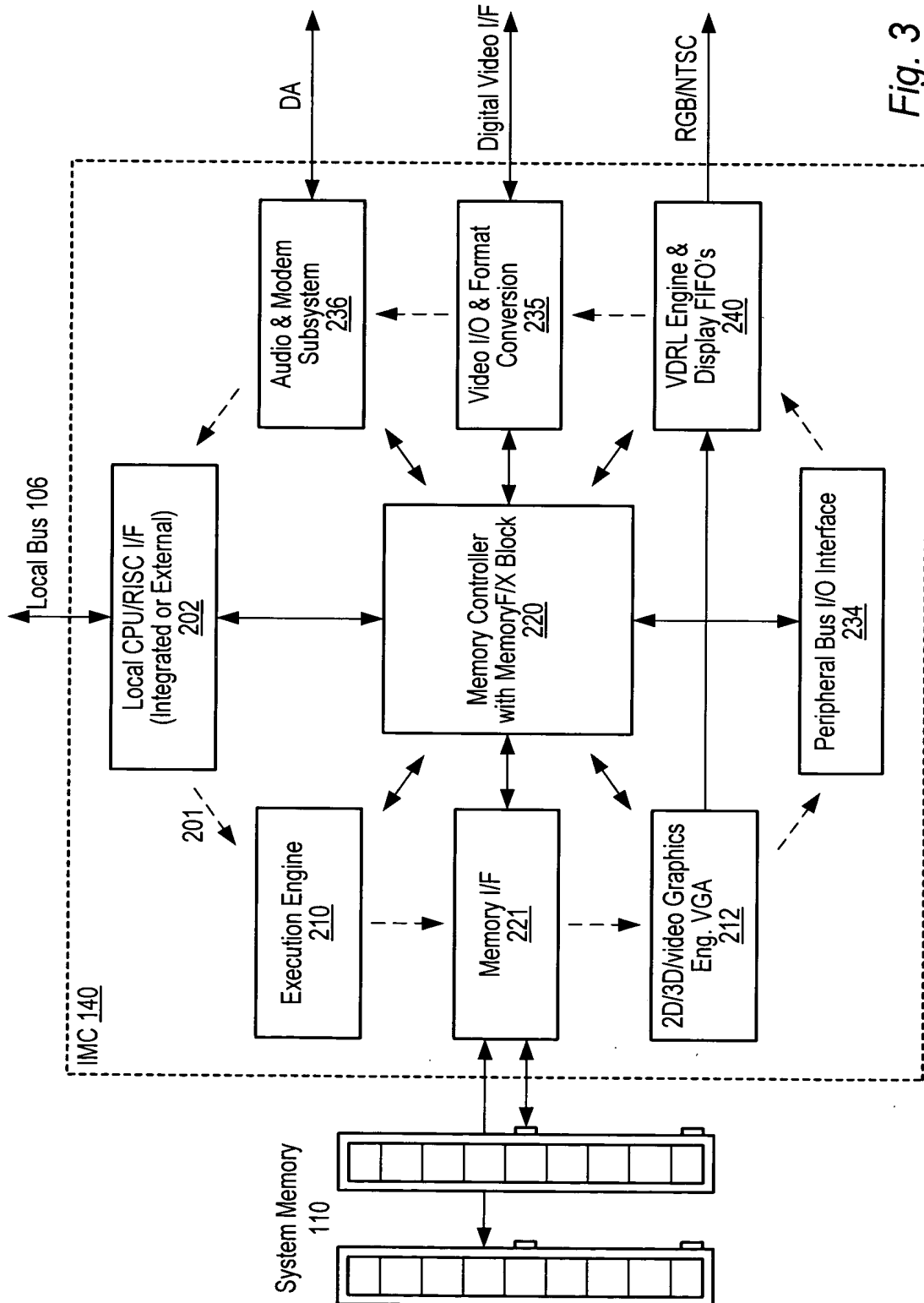
Key:
C - Command
A - Address
T - Tag
L - Length
Ty - Type
S - Status

Fig. 5

Input Data Byte 0 → Entry 0 → Entry 1 → ~ → Entry A → ~ → Entry n-1

Fig. 6A
(Prior Art)

Input Data Byte 0 → Entry 0 → Entry 4 → ~ → Entry A → ~ → Entry n-4

Input Data Byte 0 → Entry 1 → Entry 5 → ~ → Entry B → ~ → Entry n-3

Input Data Byte 0 → Entry 2 → Entry 6 → ~ → Entry C → ~ → Entry n-2

Input Data Byte 0 → Entry 3 → Entry 7 → ~ → Entry D → ~ → Entry n-1

Fig. 6B
(New Art)

```
┌─────────────────────────────────┐
│      Parallel Compression       │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│      Maintain a history table   │
│        comprising entries       │
│               402               │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│  Maintain a current count of prior │
│    matches for each entry in the   │
│          history table          │
│               404               │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│      Receive Uncompressed data  │
│               406               │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│  Compare a plurality of symbols with │
│     each entry in the history table  │
│               408               │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│ Determine match information for each │
│   of the plurality of symbols based on │
│  the current count and compare results │
│               410               │
└─────────────────────────────────┘
                 │
                 ▼
┌─────────────────────────────────┐
│ Output compressed data in response │
│      to the match information   │
│               412               │
└─────────────────────────────────┘
```

Fig. 7

OIPE JC124
JAN 2 8 2003
PATENT & TRADEMARK OFFICE

Start

More Input Data? 406 — No → Flush Remaining previous Matches 480 → Stop

More Input Data? 406 — Yes → Compare each input symbol with each Window Entry 408

Add Uncompressed Symbols to History Window 474

Any Matches? 422 — No → Previous matches? 432

Any Matches? 422 — Yes → All Symbols in one match? 442

Previous matches? 432 — No → Output Each Symbol Uncompressed 436 → Reset All Counters to Zero 438

Previous matches? 432 — Yes → Output Previous Compressed Data Info 434

All Symbols in one match? 442 — Yes → Increase Counter by # of Symbols 444

All Symbols in one match? 442 — No → Previous Matches? 452

Previous Matches? 452 — Yes → Select Largest Match Including Previous Match 454 → Output Previous Compressed Data Info 456

Add Uncompressed Symbols to History Window 472

For Each Input Symbol

Symbol Included in Any Match? 462 — No → Output Symbol Uncompressed 464

Symbol Included in Any Match? 462 — Yes → Does this Match include the last symbol? 466

Does this Match include the last symbol? 466 — No → Output Compressed Data Info 468

Does this Match include the last symbol? 466 — Yes → Reset Counter to Max of Symbol Count in Match 470

Performed in parallel. Shown serially for ease of illustration

Fig. 8

Fig. 9

Fig. 10

Input Matches				New Counter Value	Output Counter	Output Mask	Reset Value
A3	B2	C1	D0				
1	1	1	1	Saved+4	Saved +4	10000	0
0	1	1	1	0	Saved+3	10001	1
1	0	1	1	1	Saved+2	10010	2
0	0	1	1	0	Saved+2	10011	2
1	1	0	1	2	Saved+1	10100	3
0	1	0	1	0	Saved+1	10101	3
1	0	0	1	1	Saved+1	10110	3
0	0	0	1	0	Saved+1	10111	3
1	1	1	0	3	Saved	11000	4
0	1	1	0	0	Saved	01111	1
1	0	1	0	1	Saved	11010	4
0	0	1	0	0	Saved	11011	4
1	1	0	0	2	Saved	11100	4
0	1	0	0	0	Saved	11101	4
1	0	0	0	1	Saved	11110	4
0	0	0	0	0	Saved	11111	4

Fig. 11
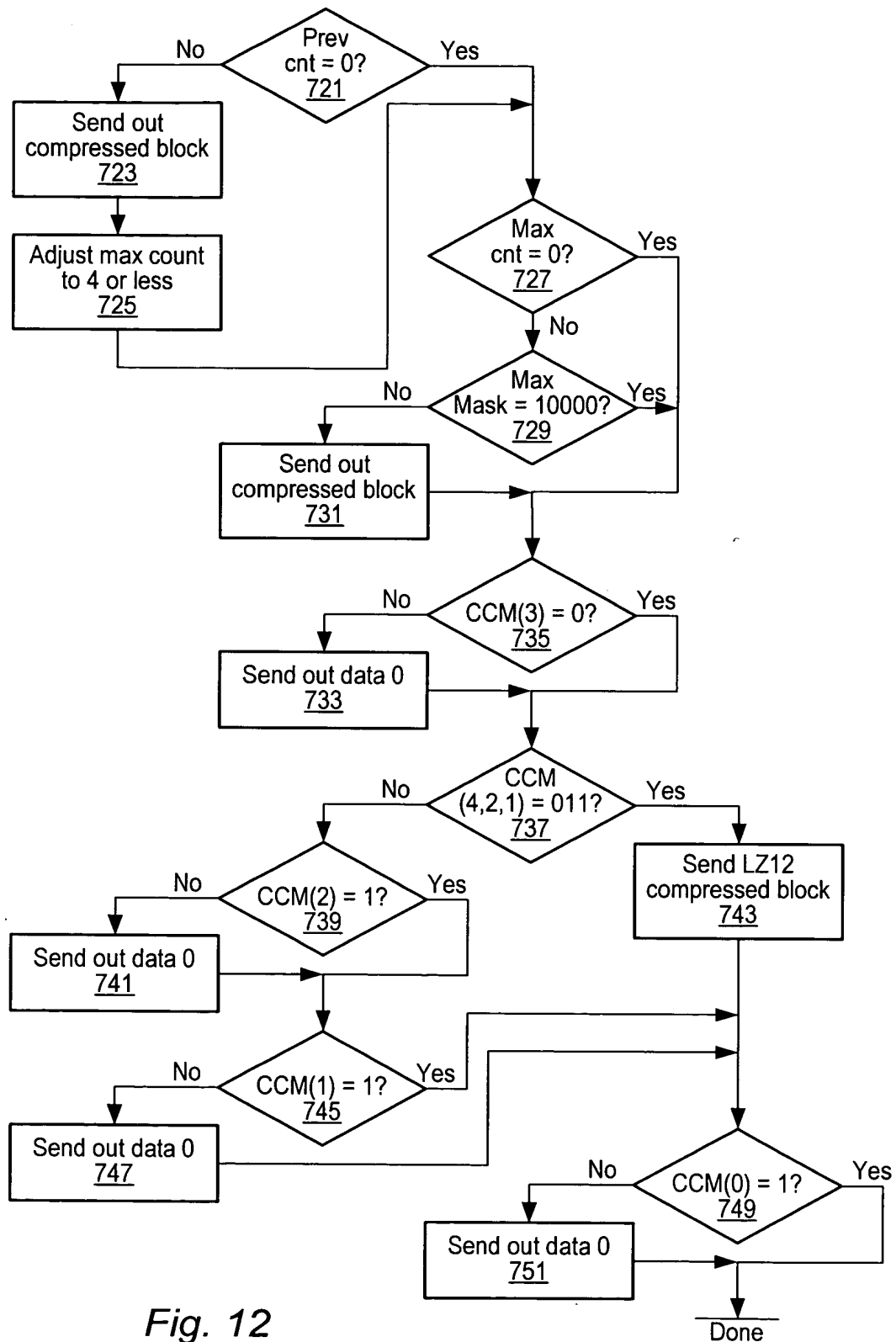
Fig. 12

	Entry	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State 0	Data	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
	Count	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input D3:0	C0F7F8F9																
	Count Out	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Mask Out	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F

Output: C0(9,3)

	Entry	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State 1	Data	C0	F7	F8	F9	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
	Count	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input D3:0	F0F1F2B5																
	Count Out	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Mask Out	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F

Output: B5

	Entry	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State 2	Data	F0	F1	F2	B5	C0	F7	F8	F9	F0	F1	F2	F3	F4	F5	F6	F7
	Count	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input D3:0	B5F7F8F9																
	Count Out	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Mask Out	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F

Output: (7,6)

	Entry	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State 3	Data	B5	F7	F8	F9	F3	F4	F5	B5	C0	F7	F8	F9	F0	F1	F2	F3
	Count	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input D3:0	F3B5C0E2																
	Count Out	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Mask Out	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F	1F

Output: (9,2)E2(6,1)

	Entry	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
State 4	Data	F3	B5	C0	E2	B5	F7	F8	F9	F3	F4	F5	B5	C0	F7	F8	F9
	Count	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

(9,2)E2B5 Alternate Output

Final Output: (7,1)

F3 Alternate Output

Fig. 13
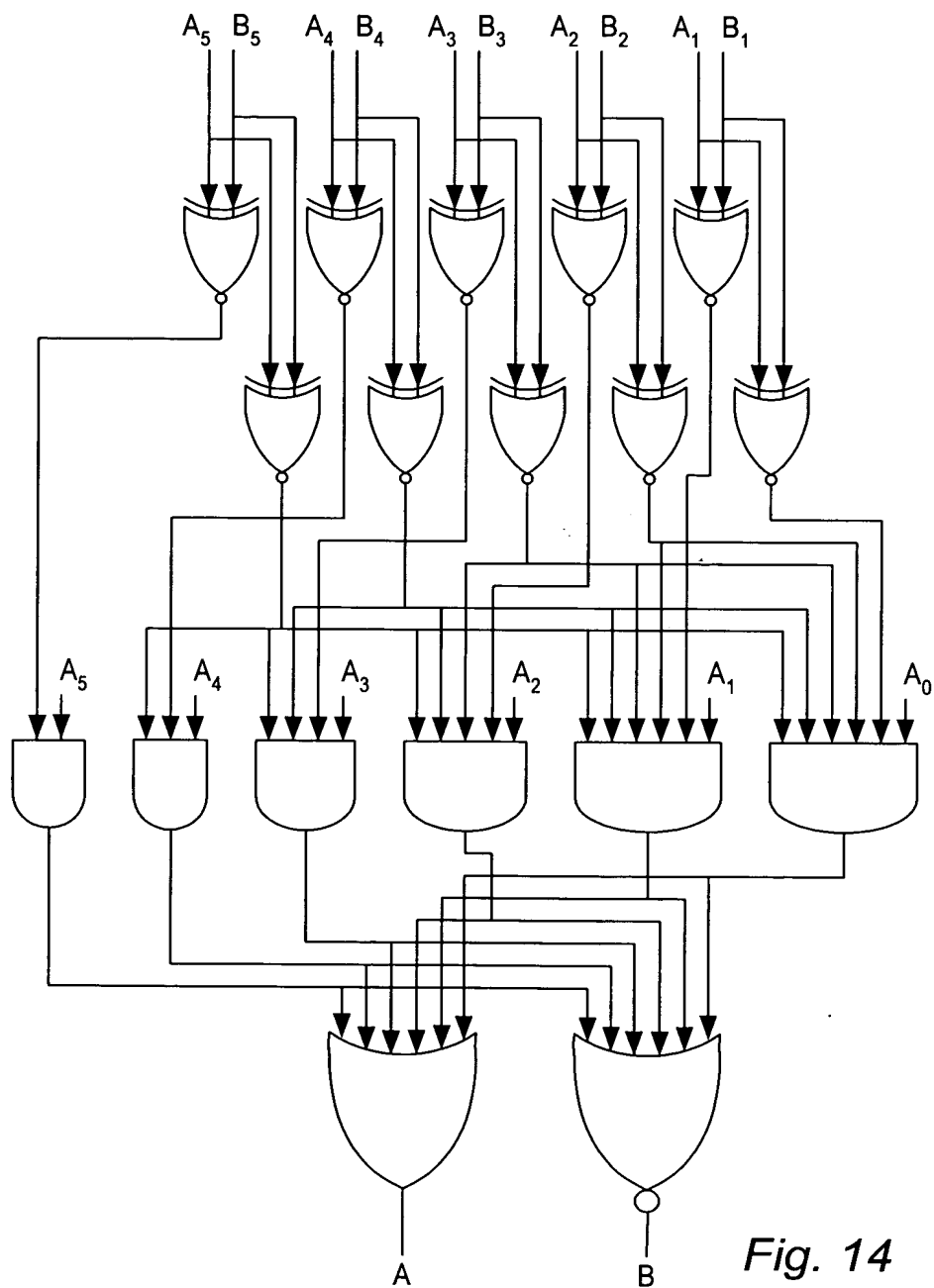
Fig. 14
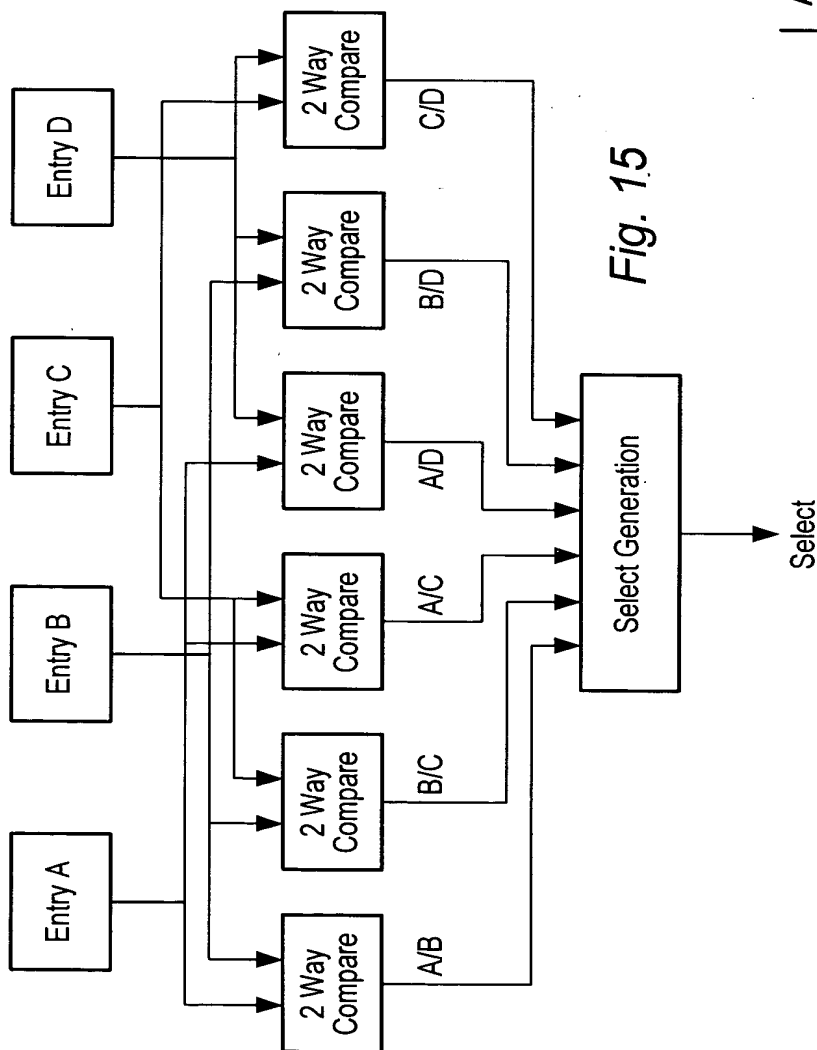
Fig. 15

A/B	B/C	C/D	D/A	A/C	B/D	Output
0	X	X	1	0	X	A
1	0	X	X	X	0	B
X	1	0	X	1	X	C
X	X	1	0	X	1	D

Fig. 16

Fig. 17

Fig. 18
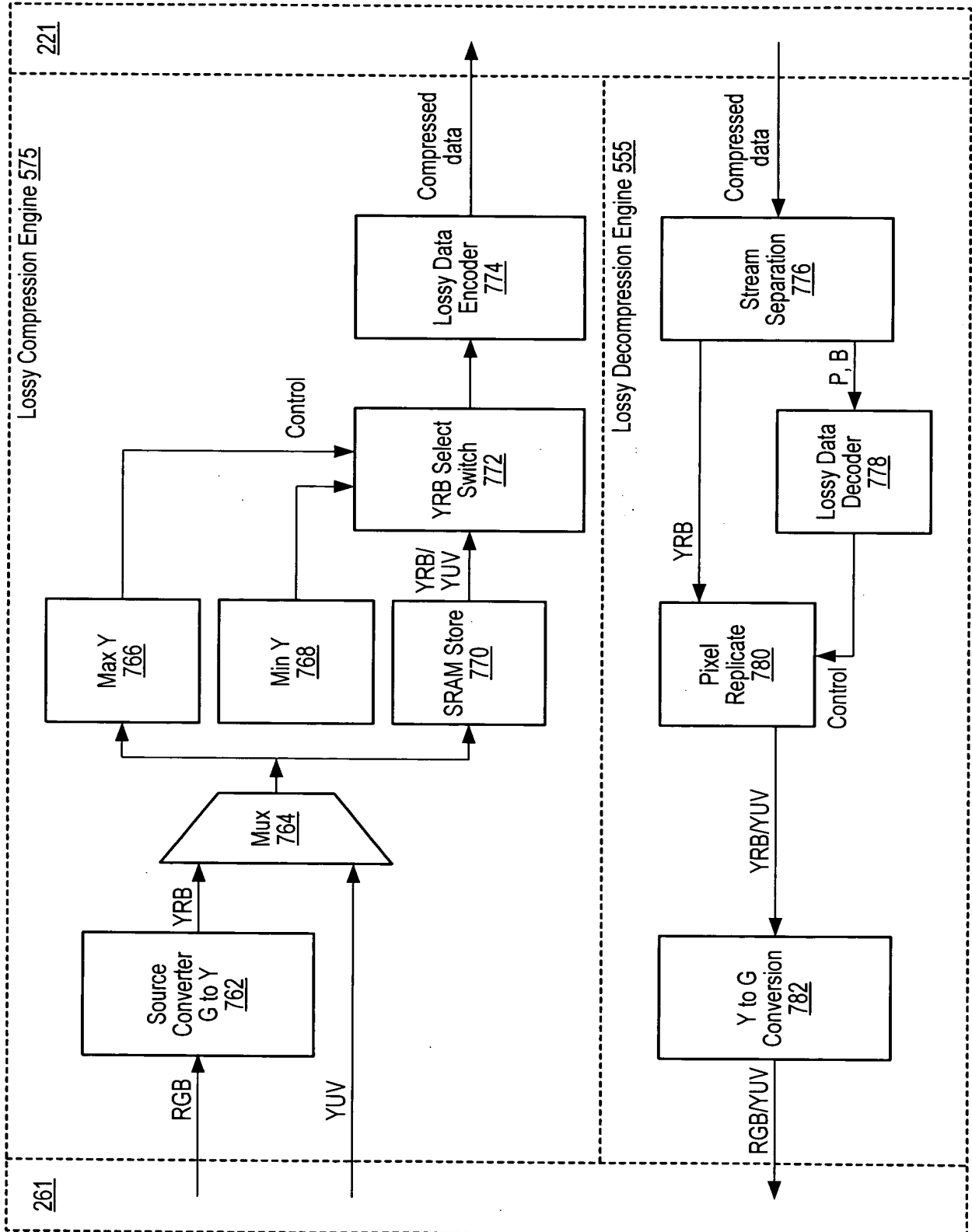
Ymax = Ymin	1 color	Ymax	Ymax	Rmax	Bmax	11		3 Bytes
		6 bits	6 bits	5 bits	5 bits	2 bits		
Ymax != Ymin	2 colors	Ymax	Ymin	Rmax	Rmin	Bmin	P bits	6 Bytes
		6 bits	6 bits	5 bits	5 bits	5 bits	16 bits	
Ymax != Ymin	>2 colors	Ymax	Ymin	Rmax	Rmin	Bmin	P bits	8 Bytes
		6 bits	6 bits	5 bits	5 bits	5 bits	32 bits	

Fig. 19

Ymax = Ymin	Amax = Amin = 0x00	1 color	Ymax	Ymax	Rmax	Bmax	00				3 Bytes
			6 bits	6 bits	5 bits	5 bits	2 bits				
Ymax = Ymin	Amax = Amin = 0xFF	1 color	Ymax	Ymax	Rmax	Bmax	11				3 Bytes
			6 bits	6 bits	5 bits	5 bits	2 bits				
Ymax = Ymin	Amax = Amin != 00 or FF	1 color	Ymax	Ymax	Rmax	Bmax	01	Amax	Amin		4/5 Bytes
			6 bits	6 bits	5 bits	5 bits	2 bits	4/8 bits	4/8 bits		
Ymax = Ymin	Amax != Amin	1 color	Ymax	Ymax	Rmax	Bmax	01	Amax	Amin	P bits	6/7 Bytes
	2 Alphas		6 bits	6 bits	5 bits	5 bits	2 bits	4/8 bits	4/8 bits	16 bits	
Ymax = Ymin	Amax != Amin	1 color	Ymax	Ymax	Rmax	Bmax	10	Amax	Amin	P bits	8/9 Bytes
	>2 Alphas		6 bits	6 bits	5 bits	5 bits	2 bits	4/8 bits	4/8 bits	32 bits	
Ymax != Ymin	X	2 colors	Ymax	Ymin	Rmax	Rmin	Bmin	Amax	Amin	P bits	7/8 Bytes
			6 bits	6 bits	5 bits	5 bits	5 bits	4/8 bits	4/8 bits	16 bits	
Ymax != Ymin	X	>2 colors	Ymax	Ymin	Rmax	Rmin	Bmin	Amax	Amin	P bits	9/10 Bytes
			6 bits	6 bits	5 bits	5 bits	5 bits	4/8 bits	4/8 bits	32 bits	

Fig. 20

Memory Bus

Compressed Data

Output Format Switch 586

SRAM Store 581

SRAM Store 582

Array Compare Unit 584

Error

Loss-Less Parallel Compression Engine 570

Lossy Compression Engine 575

Input Switch 261

INPUT Bus

Original Source 120

Output Format Switch 588

Decompressed Data

Compressed Data

Lossy Decompression Engine 555

Loss-Less Parallel Decompression Engine 550

— — — Primary compression
— · · — Intermediate
· · · · · Decompression
· · · · · · Decompression

Fig. 21

```
┌─────────────────────────────────┐
│   Compression of received data   │
└─────────────────────────────────┘
                  │
                  ▼
┌─────────────────────────────────┐
│     Receive uncompressed data    │
│               802                │
└─────────────────────────────────┘
                  │
                  ▼
┌─────────────────────────────────┐
│    Determine a compression mode  │
│           of the data            │
│               804                │
└─────────────────────────────────┘
                  │
                  ▼
┌─────────────────────────────────┐
│       Selectively compress the   │
│   uncompressed data according to the │
│          compression mode        │
│               806                │
└─────────────────────────────────┘
                  │
                  ▼
┌─────────────────────────────────┐
│  Store the data in the memory; store │
│  compression mode information with   │
│              the data            │
│               808                │
└─────────────────────────────────┘
```

Fig. 22

```
┌─────────────────────────────┐
│    Access compressed data    │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│  Receive a request for the data  │
│             812              │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│  Access the data from the memory │
│             814              │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│   Determine a compression mode   │
│          for the data         │
│             816              │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│  Selectively decompress the data │
│ according to the compression mode│
│             818              │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│   Provide the data in response   │
│         to the request        │
│             820              │
└─────────────────────────────┘
```

Fig. 23

Fig. 24

Memory Allocation Fields

Compressed Address Translation Table (CATT)-128 Entry Design Limit				
Starting Addr	Ending Addr	Type	Data Ptr	OAT Ptr
20 bits	20 bits	4 bits	20 bits	8 bits
4GB Addressability		Compressed		
4K Boundry	4K Boundry	Blk Size	4K Boundry	4K Boundry

Overflow Address Table (OAT)-256 Entry Max			
Overflow Ptr	Next Block Ptr	Next OAT Ptr	Next OAT Valid
20 bits	24 bits	8 bits	1 bit
4 GB Addressability		Points to next entry	
4K Boundry		in this table	

Initial Header Description

Value	# of bits	Meaning
0	1	Last Block/Unused
10 A (20 bits)	22	The next block is at offset A in the Overflow Area
11 IA(8+20 bits)	30	The next block is at offset A in the Overflow Area of OAT entry I

Overflow Header Description

Value	# of bits	Meaning
00	2	Last Block/Unused
01	2	The next block follows physically after this one
10A (8 bits)	10	The next block is A blocks before this one (or after?)
110A (20 bits)	23	The next block is at offset A in the Overflow Area
111 IA (8+20 bits)	31	The next block is at offset A in the Overflow Area of OAT entry I

Fig. 25

```
                ╭─────────────────────────╮
                │    @Memory Allocation    │
                │    (Initialized CATT)    │
                │          2709            │
                ╰─────────────────────────╯
                            │
                            ▼
            ┌─────────────────────────────────┐
            │        Allocate CATT Entry       │
            │              2711                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │        Arrange entry order       │
            │ (if required based on start/end address) │
            │              2713                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │      Set the compression type    │
            │              2715                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │  Allocate% of the requested memory │
            │ (Based on block size and compression type) │
            │              2717                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │ Set the data pointer to start at the initial block │
            │            in the CATT           │
            │              2719                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │  Allocate Overflow Address memory (OAM) │
            │    (Set by  IMC driver  or BIOS) │
            │   Typically 1/8th original data size │
            │              2721                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │  Initialize the OAT pointer in the CATT │
            │              2723                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │ Initialize allocated memory headers to zero │
            │              2725                │
            └─────────────────────────────────┘
                            │
                            ▼
            ┌─────────────────────────────────┐
            │  Initialize OAT entries, set overflow │
            │           pointer in OAT         │
            │              2727                │
            └─────────────────────────────────┘
```

Fig. 26

@Compressed
Memory Store (Addr N)
2749

Check
Cache for Hit
(While searching CATT)
2731

Yes

No

Calculate Initial Address I, continue to
compress data (Validate present entry)
2733

H = Header
from Address I
2759

Remaining
Compressed block
>Block Size?
2735

Yes

Get Next
Address (U=f(H))
2799

No

Store block of Compressed data at
address I with header=LAST
2737

Store block of Compressed data at
address I with header(U) Set I=U
2739

LAST=H?
2741

No

H2=Header from Address H Set
header of H to UNUSED Set H=f(H2)
2745

Yes

Store uncompressed block and header
at Add N in cache and set most
recently modified bit for cache
2743

Done

Fig. 27

```
        ┌──────────────────┐
        │  @Memory Fetch   │
        │  (address N)     │
        │     2759         │
        └──────────────────┘
                 │
                 ▼
         ╱─────────────╲
        ╱    Search      ╲  Yes    ┌──────────────────────────┐
       ⟨ Cache Directory HIT? ⟩────▶│  Deliver data from cache │
        ╲     2751       ╱          │          2752            │
         ╲─────────────╱           └──────────────────────────┘
                 │ No                          │
                 ▼                             │
   ┌────────────────────────────────┐         │
   │ Solve for initial address (I) = │         │
   │ (Matching CATT start address -  │         │
   │ N)/X (X is based on data type)  │         │
   │            2753                 │         │
   └────────────────────────────────┘         │
                 │                             │
                 ▼                             │
   ┌────────────────────────────────┐         │
   │ Fetch from memory               │         │
   │ compressed block I              │         │
   │            2755                 │         │
   └────────────────────────────────┘         │
                 │                             │
                 ▼                             │
   ┌────────────────────────────────┐         │
   │ Strip the Header bits, and      │         │
   │ decompress the remaining data   │         │
   │ from I                          │         │
   │            2757                 │         │
   └────────────────────────────────┘         │
                 │                             │
                 ▼                             │
         ╱─────────────╲                ┌─────────────────────────────┐
        ╱   LAST Block?  ╲   No          │ Fetch the overflow block    │
       ⟨     2761         ⟩─────────────▶│ using the overflow pointer  │
        ╲               ╱                │ as base and the header as   │
         ╲─────────────╱                 │ offset       2754           │
                 │ Yes                   └─────────────────────────────┘
                 │                                    │
                 │                                    ▼
                 │                       ┌─────────────────────────────┐
                 │                       │ Read compressed overflow    │
                 │                       │ blocks                      │
                 │                       │            2756             │
                 │                       └─────────────────────────────┘
                 │                                    │
                 │                                    ▼
                 │                       ┌─────────────────────────────┐
                 │                       │ Send decompressed data to   │
                 │                       │ requesting agent            │
                 │                       │            2765             │
                 │                       └─────────────────────────────┘
                 ▼
   ┌────────────────────────────────┐
   │ Update Cache, invalidate LRU,   │
   │ mark new block as MRU           │
   │            2769                 │
   └────────────────────────────────┘
                 │
                 ▼
      Completed compressed
       read operation
```

Fig. 28

Get next address (R=f(H)) 2799

LAST Block? 2791

Calculate next block address pointer A = f(Header N) 2702

Read Overflow Header @R 2707

Does header indicated Unused ? 2704

Next address pointer = R + block size 2706B

Store resulting pointer in next address pointer 2708

Done

R = R+Block Size 2706A

Is next block OAT entry full? 2793

New Overflow pointer from OAT @next pointer 2795

Is next overflow pointer valid? 2797

Allocate new block of Overflow memory and OAT entry 2701

Set new OAT pointer and valid bit 2703

Calculate new overflow block address (R)= f(OvflPntr, offset, block size) 2705

No

Yes

Yes

No

Yes

No

Yes

No

Fig. 29

Uncomp Block Bytes	Type	Initial Block Size Bytes	Overflow Block Size Bytes	Max Comp Ratio (X:1)	Initial Allocation	Header w/o OF	Header w/ OF Non-Frag	Header w/ OF Fragmented
4096	8	256	64	16	6%	0.0%	0.4%	4.1%
2048	7	128	64	16	6%	0.1%	0.5%	4.2%
1024	6	64	64	16	6%	0.2%	0.6%	4.3%
512	5	64	64	8	13%	0.2%	0.9%	4.3%
256	4	64	64	4	25%	0.2%	1.4%	4.3%
128	3	32	32	4	25%	0.4%	2.8%	8.8%
64	2	32	16	2	50%	0.4%	5.1%	13.6%
32	1	32	8	1	100%	0.4%	8.9%	11.5%

Fig. 30

— 2530

1) Read compressed block from memory

2) Decompress to L3 data cache

3) Store Position tag (@F) in table

4) Re - compress block out of order

5) Attach header with tag (@F)

6) Write compressed block back to memory

Fig. 31

— 2510

Compression block (256bytes)

New read address

| Header | A | B | C | D | E | F | G | H |

Intermediate Block Segment (32bytes)

Original Order*

— 2520

* Header tag field is '0' indicating in original order

| Header' | F | G | H | A | B | C | D | E |

New Compressed Order**

** Header tag field is '5' indicating out of order

Bytes Compressed	Flag	Index	Count	Data	Bits Used
0	0	-	-	8b	9
1	10	6b	-	-	8
2	1100	6b	-	-	10
3	1101	6b	-	-	10
4	1110	6b	-	-	10
5	1111000	6b	-	-	13
6	1111001	6b	-	-	13
7	1111010	6b	-	-	13
8	1111011	6b	-	-	13
9	1111100	6b	-	-	13
10	1111101	6b	-	-	13
11	1111110	6b	-	-	13
>11	1111111	6b	12b	-	25

Fig. 32

Input Data 8 Bytes

25501

Stage 1: Initial Input Selector and Byte Counter

Start Count		Index		Index Valid	Data Valid	Data Bytes	
0	... 7	0	... 7			0	... 7

25503

↓5 ↓5 ↓6 ↓6 ↓8 ↓8 ↓8 ... ↓8

Pipe Register (168 bits)

25505

↓5 ↓5 ↓6 ↓6 ↓8 ↓8

Stage 2: Calculate Initial Selects and Overflows

Partial Selects and Overflows
0 15

↓8   ↓8

25507

↓8   ↓8

Pipe Register (144 bits)

25509

↓8   ↓8

Stage 3: Calculate Final Selects

Final Selects
0 15

↓8   ↓8

25511

↓7   ↓7

Pipe Register (128 bits)

25513

↓8   ↓8   ↓8 ↓8

Stage 4: Data Selection

Data Out		Data Valids	History Window
0 15			

↓8   ↓8   ↓16

Fig. 33

Fig. 34

Fig. 35